

IDC DOCUMENTATION

**Message
Subsystem
Software
User Manual**



Notice

This document was published November 2000 by the Monitoring Systems Operation of Science Applications International Corporation (SAIC) as part of the International Data Centre (IDC) Documentation. Every effort was made to ensure that the information in this document was accurate at the time of publication. However, information is subject to change.

Contributors

Michael Zatloukal, Science Applications International Corporation
David Salzberg, Science Applications International Corporation

Trademarks

BEA TUXEDO is a registered trademark of BEA Systems, Inc.
ORACLE is a registered trademark of Oracle Corporation.
SAIC is a trademark of Science Applications International Corporation.
Solaris is a registered trademark of Sun Microsystems.
SPARC is a registered trademark of Sun Microsystems.
SQL*Plus is a registered trademark of Oracle Corporation.
Sun is a registered trademark of Sun Microsystems.
UNIX is a registered trademark of UNIX System Labs, Inc.

Ordering Information

The ordering number for this document is SAIC-00/3001.

This document is cited within other IDC documents as [IDC6.5.19].

Message Subsystem Software User Manual

CONTENTS

<u>About this Document</u>	i
■ <u>PURPOSE</u>	ii
■ <u>SCOPE</u>	ii
■ <u>AUDIENCE</u>	ii
■ <u>RELATED INFORMATION</u>	ii
■ <u>USING THIS DOCUMENT</u>	iii
<u>Conventions</u>	iv
<u>Chapter 1: Introduction</u>	1
■ <u>SOFTWARE OVERVIEW</u>	2
■ <u>STATUS OF DEVELOPMENT</u>	5
■ <u>FUNCTIONALITY</u>	5
<u>Features and Capabilities</u>	6
<u>Performance Characteristics</u>	9
<u>Related Tools</u>	11
■ <u>INVENTORY</u>	11
■ <u>ENVIRONMENT AND STATES OF OPERATION</u>	14
<u>Software Environment</u>	14
<u>Normal Operational State</u>	14
<u>Contingencies/Alternate States of Operation</u>	14
<u>Chapter 2: Operational Procedures</u>	17
■ <u>SOFTWARE STARTUP</u>	18
■ <u>SOFTWARE SHUTDOWN</u>	19
■ <u>BASIC PROCEDURES</u>	21
<u>Obtaining Help</u>	21
■ <u>ADVANCED PROCEDURES</u>	21

Changing Status in msgdisc Table	22
Processing Messages without Mailer	23
Removing Problem Messages	23
Software Configuration	23
■ MAINTENANCE	24
■ SECURITY	25
Passwords	25
Marking/Storing Controlled Outputs	26
Chapter 3: Troubleshooting	27
■ MONITORING	28
Verifying Processes	28
Verifying Active Processes	28
Screening TSD	29
Verifying Auxiliary Data Processed	30
Monitoring Status in Database Tables	30
Screening Log Files	31
■ INTERPRETING ERROR MESSAGES	32
■ SOLVING COMMON PROBLEMS	33
Error Recovery	34
■ REPORTING PROBLEMS	35
Chapter 4: Installation Procedures	37
■ PREPARATION	38
Obtaining Released Software	39
Hardware Mapping	39
UNIX System	39
Firewall	40
■ EXECUTABLE FILES	40
■ CONFIGURATION DATA FILES	40
■ DATABASE	43
Accounts	43
Tables	43

<u>Initialization of LastID</u>	47
■ <u>INITIATING OPERATIONS</u>	47
■ <u>VALIDATING INSTALLATION</u>	47
<u>References</u>	49
<u>Glossary</u>	G1
<u>Index</u>	I1

Message Subsystem Software User Manual

FIGURES

<u>FIGURE 1.</u>	<u>IDC SOFTWARE CONFIGURATION HIERARCHY</u>	3
<u>FIGURE 2.</u>	<u>IDC PROCESSING FLOW</u>	4
<u>FIGURE 3.</u>	<u>INBOUND MESSAGE SUBSYSTEM FLOW</u>	8
<u>FIGURE 4.</u>	<u>OUTBOUND MESSAGE SUBSYSTEM FLOW</u>	9
<u>FIGURE 5.</u>	<u>ENTITY RELATIONSHIPS OF MESSAGE SUBSYSTEM TABLES</u>	45

Message Subsystem Software User Manual

TABLES

TABLE I:	DATA FLOW SYMBOLS	v
TABLE II:	ENTITY-RELATIONSHIP SYMBOLS	v
TABLE III:	TYPOGRAPHICAL CONVENTIONS	vi
TABLE IV:	TECHNICAL TERMS	vi
TABLE 1:	MESSAGE SUBSYSTEM REQUEST RESPONSE TIMES	10
TABLE 2:	MESSAGE SUBSYSTEM INVENTORY	11
TABLE 3:	MESSAGE PROCESSING STATES	22
TABLE 4:	MESSAGE SUBSYSTEM MAIL ALIASES	40
TABLE 5:	MESSAGE SUBSYSTEM DATABASE TABLES	44

About this Document

This chapter describes the organization and content of the document and includes the following topics:

- [Purpose](#)
- [Scope](#)
- [Audience](#)
- [Related Information](#)
- [Using this Document](#)

About this Document

PURPOSE

This document describes how to use the Message Subsystem software of the International Data Centre (IDC). The software is a computer software component (CSC) of the Data Services Computer Software Configuration Item (CSCI) and is identified as follows:

Title: Message Subsystem

Identification Number: CSC4.2

SCOPE

The manual includes instructions for setting up the software, using its features, and basic troubleshooting. This document does not describe the software's design or requirements. These topics are described in sources cited in "[Related Information](#)."

AUDIENCE

This document is intended for the first-time or occasional user of the software. However, more experienced users may find certain sections useful as a reference.

RELATED INFORMATION

The following documents complement this document:

- Software Design Description for the *Message Subsystem* [IDC7.4.2]

See [“References” on page 49](#) for a list of documents that supplement this document. The following UNIX man pages apply to the existing Message Subsystem software:

- *AutoDRM*
- *MessageAlert*
- *MessageFTP*
- *MessageGet*
- *MessageReceive*
- *MessageShip*
- *MessageStore*
- *ParseData*

USING THIS DOCUMENT

This document is part of the overall documentation architecture for the IDC. It is part of the Technical Instructions category, which provides guidance for installing, operating, and maintaining the IDC systems. This document is organized as follows:

- [Chapter 1: Introduction](#)
This chapter provides an overview of the software’s capabilities, development, and operating environment.
- [Chapter 2: Operational Procedures](#)
This chapter describes how to use the software and includes detailed procedures for startup and shutdown, basic and advanced features, security, and maintenance.
- [Chapter 3: Troubleshooting](#)
This chapter describes how to identify and correct common problems related to the software.

▼ About this Document

- [Chapter 4: Installation Procedures](#)

This chapter describes first how to prepare for installing the software, then how to install the executable files, configuration data files, and database elements. It also describes how to initiate operation and how to validate the installation.

- [References](#)

This section lists the sources cited in this document.

- [Glossary](#)

This section defines the terms, abbreviations, and acronyms used in this document.

- [Index](#)

This section lists topics and features provided in this document along with page numbers for reference.

Conventions

This document uses a variety of conventions, which are described in the following tables. [Table I](#) shows the conventions for data flow diagrams. [Table II](#) shows the conventions for entity-relationship diagrams. [Table III](#) lists typographical conventions. [Table IV](#) explains certain technical terms that are not part of the standard Glossary, which is located at the end of this document.

TABLE I: DATA FLOW SYMBOLS

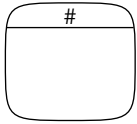


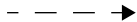




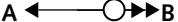






Description	Symbol
process	
external source or sink of data	
data store D = disk Db = database	
control flow	
data flow	

TABLE II: ENTITY-RELATIONSHIP SYMBOLS

Description	Symbol														
One A maps to one B.	A  B														
One A maps to zero or one B.	A  B														
One A maps to many Bs.	A  B														
One A maps to zero or many Bs.	A  B														
database table	<table border="1"> <tr> <td colspan="2">tablename</td> </tr> <tr> <td></td> <td>primary key</td> </tr> <tr> <td></td> <td>foreign key</td> </tr> <tr> <td colspan="2">attribute 1</td> </tr> <tr> <td colspan="2">attribute 2</td> </tr> <tr> <td colspan="2">...</td> </tr> <tr> <td colspan="2">attribute n</td> </tr> </table>	tablename			primary key		foreign key	attribute 1		attribute 2		...		attribute n	
tablename															
	primary key														
	foreign key														
attribute 1															
attribute 2															
...															
attribute n															

▼ About this Document

TABLE III: TYPOGRAPHICAL CONVENTIONS

Element	Font	Example
database table	bold	msgdisc
database table and attribute, when written in the dot notation		msgdisc.msgid
database attributes	<i>italics</i>	<i>msgid</i>
processes, software units, and libraries		<i>AutoDRM</i>
titles of documents		<i>Configuration of PIDC Databases</i>
computer code and output	courier	testbed 16603 MessageGet
filenames, directories, and web sites		MessageGet.par
user-defined arguments and variables used in parameter (par) files or program command lines		AUXDIR=message_path/w
text that should be typed in exactly as shown		% ps -ef grep Message

TABLE IV: TECHNICAL TERMS

Term	Description
hang	halt processing without error generation
instance	running computer program; an individual program may have multiple instances on one or more host computers
invoke	execute a program
lastid	database table containing counter values
TSD	temporary storage directory for messages received by email
TSD FTP	temporary storage directory for FTP messages

Chapter 1: Introduction

This chapter provides a general description of the software and includes the following topics:

- [Software Overview](#)
- [Status of Development](#)
- [Functionality](#)
- [Inventory](#)
- [Environment and States of Operation](#)

Chapter 1: Introduction

SOFTWARE OVERVIEW

[Figure 1](#) shows the logical organization of the IDC software. The Message Subsystem is one component of the Data Services CSCI. [Figure 2](#) shows a processing flow model of the IDC system and the relationship of the Message Subsystem to other components of the system.

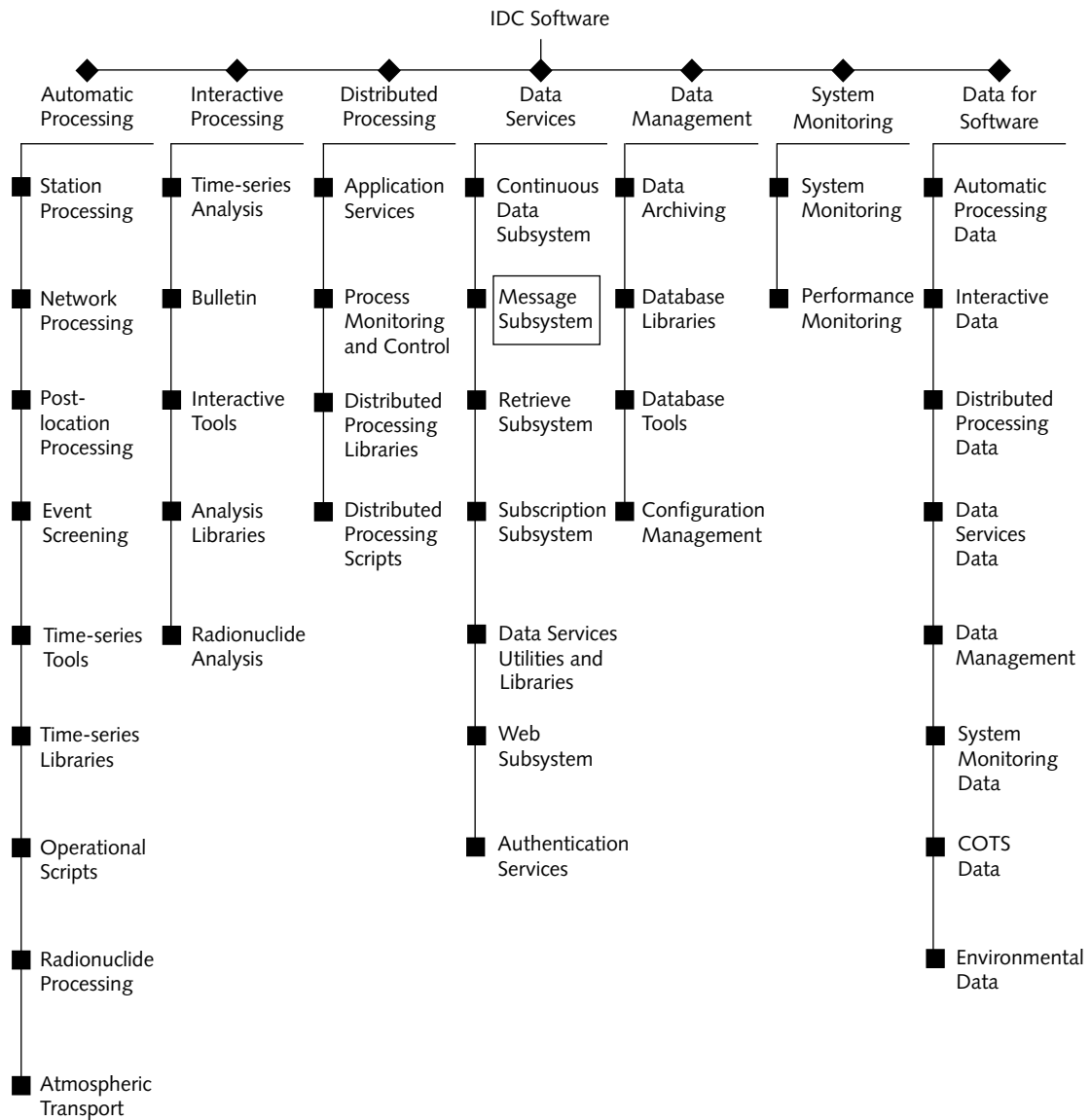


FIGURE 1. IDC SOFTWARE CONFIGURATION HIERARCHY

▼ Introduction

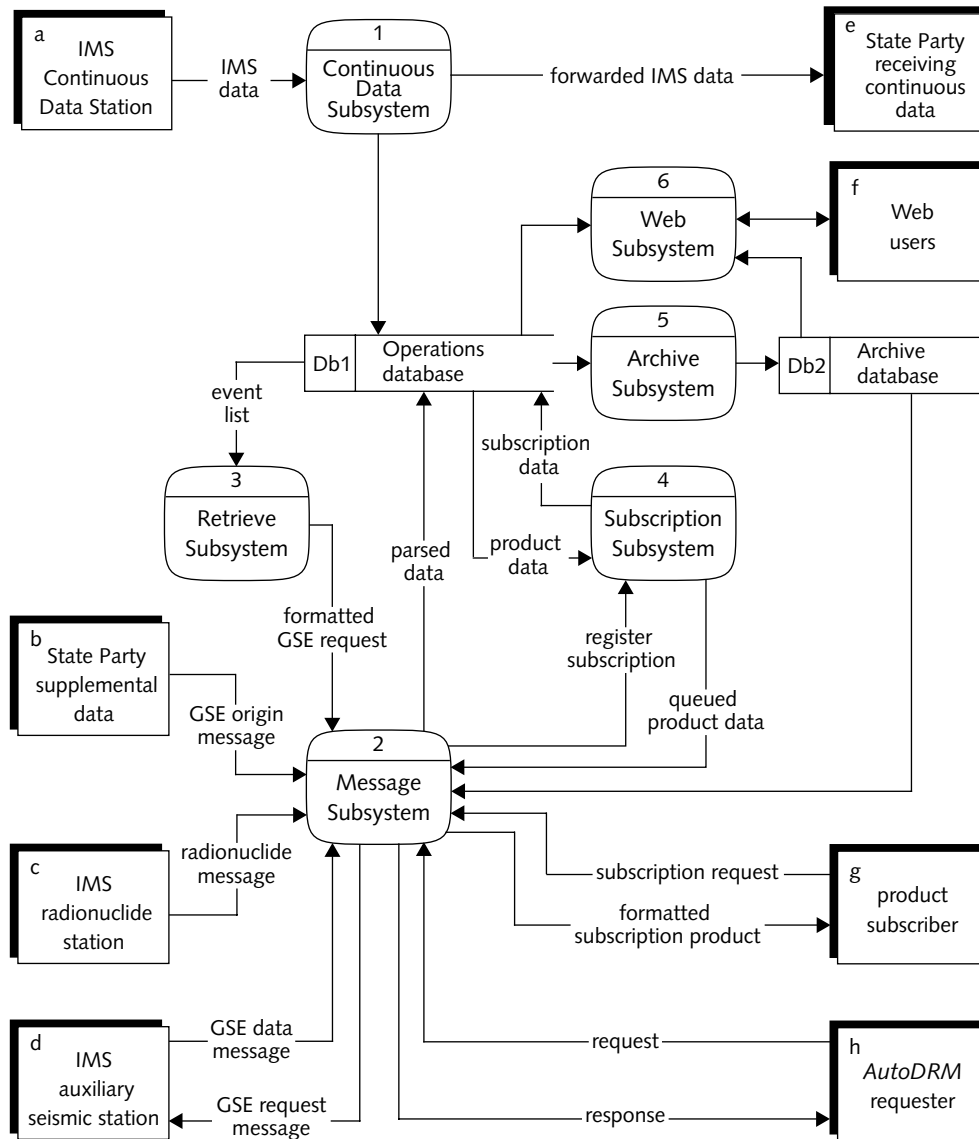


FIGURE 2. IDC PROCESSING FLOW

STATUS OF DEVELOPMENT

The majority of Message Subsystem development is almost complete. In the future, the Message Subsystem will verify users and order requests to be processed in a more deterministic method. Additional products and formats will be supported. Refer to the software design description of the *Message Subsystem* [\[IDC7.4.2\]](#).

FUNCTIONALITY

The Message Subsystem enables users to request and receive IDC products. Requests are sent using email, and the products are delivered using email or retrieved using FTP. Requests are sent in IMS 1.0 format request messages [\[IDC3.4.1Rev2\]](#), and the products are delivered or retrieved in the requested IMS format. The Retrieve Subsystem uses the Message Subsystem to send requests to auxiliary seismic stations and to receive and parse their responses. The Message Subsystem parses data messages from radionuclide stations. Incoming subscription messages are routed to the Subscription Subsystem, and subscription data messages are delivered by the Message Subsystem.

Message Subsystem users must have access to an SMTP (simple mail transfer protocol) agent. An FTP (file transfer protocol) client is needed to retrieve products too large for email delivery.

A brief description of the components of the Message Subsystem follows:

<i>AutoDRM</i>	provides automated responses to requests for data or data products
<i>MessageAlert</i>	forwards unknown message types received by the subsystem to the operator
<i>MessageFTP</i>	retrieves data made available by FTP from remote sites

▼ Introduction

<i>MessageGet</i>	queues messages based on message type and distributes them to <i>AutoDRM</i> , <i>ParseData</i> , <i>MessageFTP</i> , <i>MessageAlert</i> , or <i>ParseSubs</i> (Subscription Subsystem)
<i>MessageReceive</i>	reads and categorizes GSE 2.0 and IMS 1.0 messages from the TSD (temporary storage directory)
<i>MessageShip</i>	forwards outgoing messages to the system mailer application (<i>mailx</i>).
<i>MessageStore</i>	writes incoming messages to the TSD
<i>ParseData</i>	parses incoming data messages and stores the data in the operations database

MessageGet, *MessageReceive*, and *MessageShip* are directly invoked by the user (or from a script invoked by *cron*) at startup. *MessageStore* is invoked by *sendmail*. The remaining processes are invoked by *MessageGet*.

Sendmail and *mailx* are UNIX utilities that manage email communications independent of the user.

Refer to the software design description of the *Message Subsystem* [\[IDC7.4.2\]](#) for a description of the relationship of the software's functions with interfacing systems.

Features and Capabilities

The Message Subsystem provides an automatic method to handle data and requests using email as the primary transmission method. [Figure 3](#) and [Figure 4](#) show the flow of messages through the Message Subsystem. All externally created messages are emailed or retrieved via FTP into the IDC Message Subsystem.

MessageStore receives all incoming email messages. *MessageStore* stores the message in the TSD using a unique file name.

MessageReceive polls the TSD and FTP_TSD and parses the files, checking the message type and verifying the message format. *MessageReceive* then reads these files and creates an entry for the messages in the **msgdisc** table with *status* = RECEIVED. *MessageReceive* copies the file to a message directory and deletes the file from the TSD. *MessageReceive* runs continuously.

MessageFTP retrieves messages from a remote site via FTP and based on FTP_LOG messages received via email. The retrieved messages are placed in an FTP_TSD.

MessageGet queries the database for messages with *status* = RECEIVED. *MessageGet* forwards messages with *status* = RECEIVED to the appropriate application based on message type and subtype by forking a child process. Request messages fork *AutoDRM*. Data messages fork *ParseData*. FTP messages fork *MessageFTP*. Problem and unknown message types fork *MessageAlert*. Subscription messages fork *ParseSubs*. *MessageGet* runs continuously.

AutoDRM queries the IDC databases for the requested data and generates a response message. *AutoDRM* creates another **msgdisc** entry with *status* = NEW and a **msgdest** entry with *status* = PENDING.

MessageShip queries the database for **msgdest** entries with the *status* = PENDING, then retrieves the associated **msgdisc** record using the *msgid* key. *MessageShip* either has the message emailed or stores the message where it will be available for FTP retrieval¹. The size of the message determines which method is used. If a message is less than 100 KB² and the response method is email, *MessageShip* has the message emailed to the proper destination. Otherwise, *MessageShip* creates a compressed copy of the data response on the FTP server and creates the corresponding email message that notifies the requestor that the data is available for FTP retrieval. *MessageShip* creates new **msgdisc** and **msgdest** table entries corresponding to the FTP retrieval message. The original data response's **msgdest.transmeth** is updated to FTP while the new **msgdest.transmeth** is assigned the value email. *MessageShip* runs continuously.

1. The PIDC has configured FTP for users to retrieve large messages.
2. This value is configured. The values at the PIDC are set to 100 KB for data messages in response to requests and to a default of 500 KB for subscription data messages.

▼ Introduction

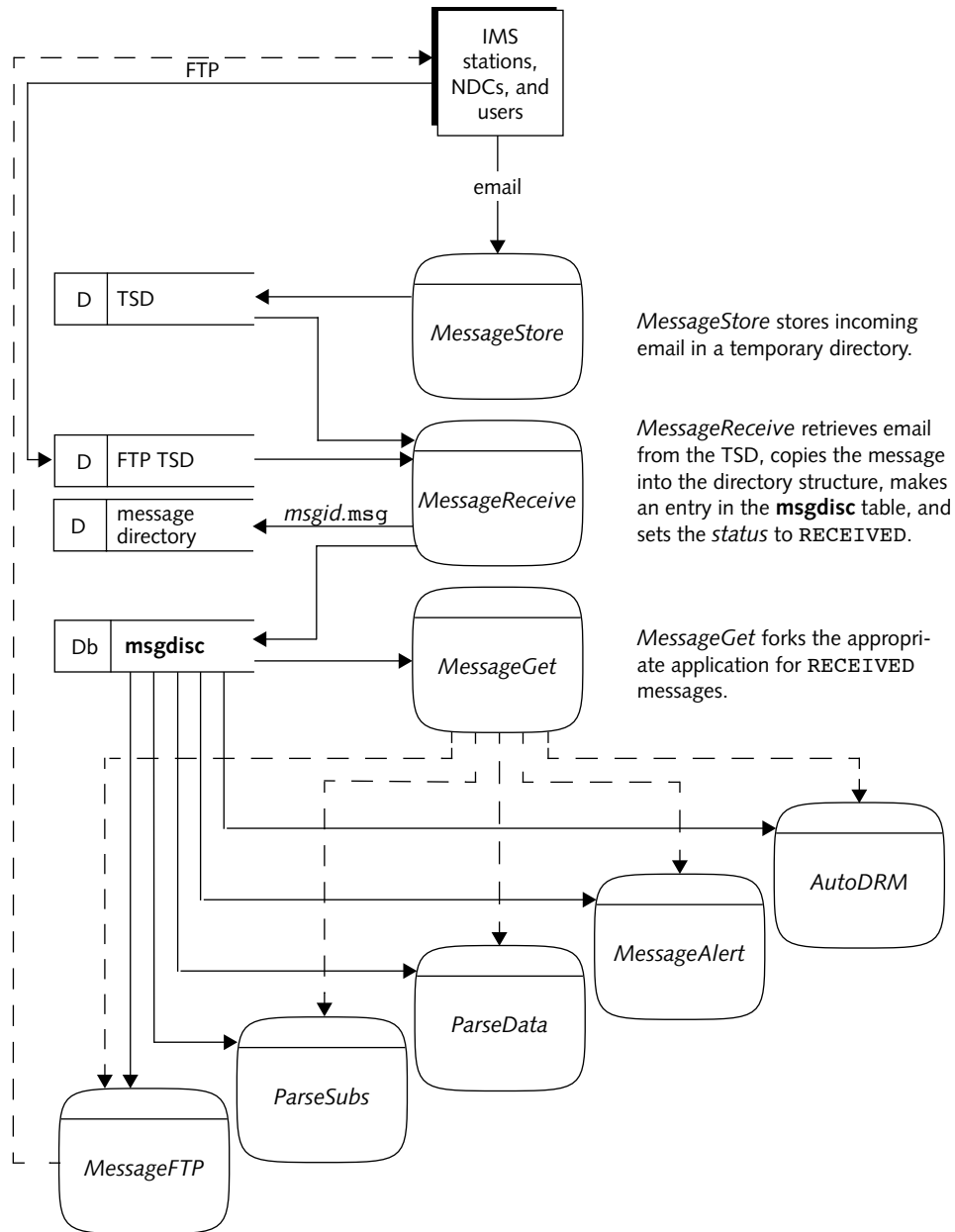


FIGURE 3. INBOUND MESSAGE SUBSYSTEM FLOW

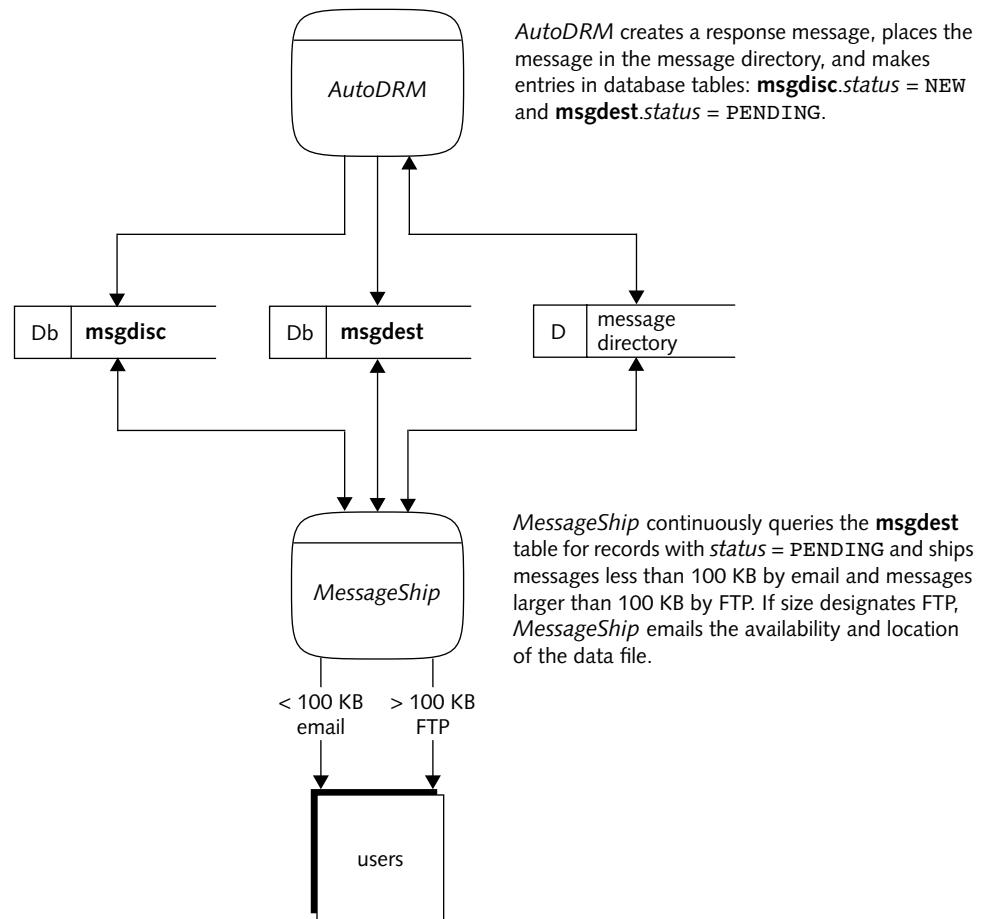


FIGURE 4. OUTBOUND MESSAGE SUBSYSTEM FLOW

Performance Characteristics

Typical Message Subsystem response statistics from PIDC operations for June 1999 are given in [Table 1](#) as an example of subsystem performance. These statistics refer to the software's mean numbers of requests and response times.

▼ Introduction

In June 1999, the Message Subsystem processed an average of 377 requests per day. The largest number of requests processed in a single day was 2,603. All messages with response times longer than an hour were requests for waveform data.

TABLE 1: MESSAGE SUBSYSTEM REQUEST RESPONSE TIMES

Length of Time for Response	Messages Processed during June, 1999	Percent of Messages
< 1 minute	2,057	18.18
< 1 hour	3,098	27.39
< 1 day	2,877	25.43
> 1 day	3,280	29.00
	11,312	100.00

Processing time is affected by the type of request, numbers of requests, and the availability of hardware. Waveform requests for archived data take longer to process. Access to archived waveform data on the mass store device (tape) includes the time required to load and tension tape. Response time is affected by the availability of the mass store device. The subsystem load (number of requests queued) affects the mean processing time. The number of simultaneous processes for request messages can be adjusted by modifying values in `MessageGet.par` to improve system performance. Current configuration specifies two *AutoDRM* processes each for long (greater than one day) non-waveform, short (less than one day) non-waveform, recent (files on disk loops) waveform, and standard (files on the mass storage device) waveform request messages.

The Message Subsystem is subject to messages created by external users and it is possible for improperly formatted or non-standardized messages to expose an error in the software.

Related Tools

The script *keep_msg_alive* is used to provide near continual operation of the Message Subsystem. This script is described in [“Maintenance” on page 24](#) and [“Installation Procedures” on page 37](#).

The CSCI 3 GUI tools *WorkFlow* and *RequestFlow* are helpful for monitoring operation of the Message Subsystem. These tools are used to verify that auxiliary data are being received. These programs provide a graphical indication of interval processing and data being written to database tables.

INVENTORY

Items needed to operate the Message Subsystem are listed in [Table 2](#).

TABLE 2: MESSAGE SUBSYSTEM INVENTORY

Item	Type
<i>AutoDRM</i>	executable
<i>MessageAlert</i>	executable
<i>MessageFTP</i>	executable
<i>MessageGet</i>	executable
<i>MessageReceive</i>	executable
<i>MessageShip</i>	executable
<i>MessageStore</i>	executable
<i>ParseData</i>	executable
<i>AutoDRM.par</i>	parameter file
<i>AutoDRM_r.par</i>	parameter file
<i>MessageAlert.par</i>	parameter file
<i>MessageFTP.par</i>	parameter file
<i>MessageGet.par</i>	parameter file

▼ Introduction

TABLE 2: MESSAGE SUBSYSTEM INVENTORY (CONTINUED)

Item	Type
MessageReceive.par	parameter file
MessageShip.par	parameter file
MessageStore.par	parameter file
ParseData.par	parameter file
shared.par	parameter file
msgs.par	parameter file
AutoDRM.hlp	help message file
invalid_mail	invalid mail file
offline_mail	off-line mail file
<i>libdbq</i>	library
<i>libgsefmt</i>	library
<i>libgsemsg</i>	library
<i>libgsewf</i>	library
affiliation	database table (used)
amplitude	database table (used)
arrival	database table (used)
assoc	database table (used)
datauser	database table (owned)
detection	database table (used)
ftpfailed	database table (owned)
ftplogin	database table (owned)
instrument	database table (used)
lastid	database table (used)
msgaux	database table (owned)
msgdatatype	database table (owned)

TABLE 2: MESSAGE SUBSYSTEM INVENTORY (CONTINUED)

Item	Type
msgdest	database table (owned)
msgdisc	database table (owned)
netmag	database table (used)
origaux	database table (used)
origerr	database table (used)
origin	database table (used)
outage	database table (owned)
remark	database table (used)
sensor	database table (used)
site	database table (used)
sitechan	database table (used)
stamag	database table (used)
wfdisc	database table (used)
wftag	database table (used)
xtag	database table (owned)
<i>sendmail</i>	system mailer
<i>keep_msg_alive.sh</i>	script

ENVIRONMENT AND STATES OF OPERATION

Software Environment

The Message Subsystem is designed to run on a Sun Microsystems workstation such as the SPARCstation 20/612. The hardware is configured with a minimum of 64 MB of memory and 2 GB of magnetic disk space. The Message Subsystem requires Solaris 2.6, ORACLE Client 7, the public domain utility *gzip*, and access to the UNIX *compress* utility.

The Message Subsystem requires database access and access to a SMTP mail agent (*mailx*). *WorkFlow* or *RequestFlow* programs aid in monitoring the Message Subsystem.

Normal Operational State

The Message Subsystem is designed to run continuously. It is either invoked by the user or by *cron* during installation of the release. If a Message Subsystem program exits or hangs, the subsystem or the program will need to be restarted by either the operator or the *keep_msg_alive* script.

Contingencies/Alternate States of Operation

Messages can be processed by components of the Message Subsystem in an alternate state of operation. These methods are impractical for normal operational use and are useful during diagnostic and testing activities.

The Message Subsystem can process messages (containing email headers) that are moved directly into the TSD while the subsystem is in operation. This feature is useful for processing messages without the use of a system mailer. Some components (for example *AutoDRM*, *MessageAlert*, *ParseData*, and *MessageFTP*) can be invoked from the command line to process individual messages when the sub-

system as a whole is not in operation. This is primarily useful in diagnostic activities. Refer to [“Chapter 3: Troubleshooting” on page 27](#) for more information about these methods.

Chapter 2: Operational Procedures

This chapter provides instructions for using the software and includes the following topics:

- [Software Startup](#)
- [Software Shutdown](#)
- [Basic Procedures](#)
- [Advanced Procedures](#)
- [Maintenance](#)
- [Security](#)

Chapter 2: Operational Procedures

SOFTWARE STARTUP

The Message Subsystem runs continuously in a stand-alone mode independent from Tuxedo control. The Message Subsystem components can be started manually by a user or automatically by a script called from *cron*.

To start the Message Subsystem, the programs *MessageGet*, *MessageReceive*, and *MessageShip* are invoked. *MessageGet* invokes the remaining Message Subsystem programs by forking child processes with the exception of *MessageStore*. *MessageStore* is invoked by *sendmail*.

Use the following steps to start the Message Subsystem manually:

1. Log onto the machine designated as the Message Subsystem host, and access the UNIX user or user group required for Message Subsystem operation (the Message Subsystem process will have the UNIX group permissions of the user who invokes the programs).
2. Execute *MessageReceive*, *MessageShip*, and *MessageGet* by using the par file for each program as the command line argument as follows:

```
MessageReceive par=/cmss/config/app_config/messages/\
MessageReceive/MessageReceive.par &
MessageGet par=/cmss/config/app_config/messages/\
MessageGet/MessageGet.par &
MessageShip par=/cmss/config/app_config/messages/\
MessageShip/MessageShip.par &
```

3. The Message Subsystem should now be running. Verify processes using the `ps -ef | grep message` command.

The Message Subsystem programs can be invoked automatically using the UNIX *cron* utility to start the *keep_msg_alive* script. This method is recommended for optimal performance of the subsystem. To use this method:

1. Edit *cron* to execute the script *keep_msg_alive* located in */cmss/scripts*. For optimal performance of the Message Subsystem, *cron* should execute *keep_msg_alive* at frequent intervals (for example, 5–15 min.).

keep_msg_alive checks that the Message Subsystem programs are not running before invoking them. Because *cron* runs *keep_msg_alive* periodically, if any or all three of the programs exit, *keep_msg_alive* will restart them.

After the user or *keep_msg_alive* invokes the Message Subsystem, check to ensure the processes are running. Processes for *MessageGet*, *MessageReceive*, and *MessageShip* should be displayed.

```
% ps -ef | grep Message
testbed 16603 MessageGet par=/cmss/config/app_config/
      messages/MessageGet/MessageGet.par
testbed 16608 1 0 21:15:01 MessageShip par=/cmss/config/app_config/
      messages/MessageShip/MessageShip.par
testbed 16598 1 0 21:15:00 MessageReceive par=/cmss/config/
      app_config/messages/MessageReceive/MessageReceive.par
```

Refer to [“Monitoring” on page 28](#) for more information about verifying Message Subsystem operation.

SOFTWARE SHUTDOWN

The Message Subsystem can be shut down gradually to allow current processing to complete, or it can be shut down immediately. Some cleanup is required if the latter method is used.

If *keep_msg_alive* is used to keep the subsystem running, first edit the *crontab* entry to prevent *keep_msg_alive* from restarting the Message Subsystem.

If time allows, shut down the Message Subsystem gradually to allow its current processing to complete. Use the following steps to gradually terminate subsystem operations:

▼ Operational Procedures

1. Terminate *MessageReceive* by creating a file named `.end` in the TSD.¹ Check the processes until *MessageReceive* is no longer running.
2. Allow *MessageGet* to forward remaining messages to their appropriate destinations before killing the process. To determine when *MessageGet* is done, query to see if any messages have *status* = RECEIVED or QUEUED in the **msgdisc** table. When no messages have these statuses, *MessageGet* can be safely killed. From the command line, enter the *kill* command using the process identifier (ID) of *MessageGet* as the argument (for example, `kill 248`).
3. Allow *AutoDRM*, *ParseData* and *MessageAlert* to finish processing messages. Query all the **msgdisc** records where *status* is not equal to DONE, PARSED, or FAILED. After this is accomplished, *AutoDRM*, *ParseData*, and *MessageAlert* can be killed safely.
4. *MessageShip* can be killed after all messages in the **msgdest** table have *status* = DELIVERED.

The Message Subsystem is shut down.

If the Message Subsystem needs to be shut down immediately, use the following steps:

1. Identify the process ID numbers of *MessageAlert*, *MessageGet*, *MessageReceive*, *MessageShip*, *ParseData*, and *AutoDRM* as follows:

```
% ps -ef | grep Message
testbed 16603 MessageGet
testbed 16608 MessageShip
testbed 16598 MessageReceive
testbed 16611 MessageAlert
% ps -ef | grep AutoDRM
testbed 16601 AutoDRM
% ps -ef | grep ParseData
testbed 16609 ParseData
```

1. Before the Message Subsystem is restarted again, the `.end` file in the TSD must be removed.

2. Kill these processes using the process ID numbers as the arguments to the *kill* command as follows:

```
% kill 16603 16608 16598 16601 16609 16611
```

The Message Subsystem is shut down.

When an *AutoDRM* process is killed, the **msgdisc** table entry for the message *AutoDRM* was processing will be updated to *status* = KILLED. When the Message Subsystem is restarted, query the **msgdisc** table for entries with *status* = KILLED, and update their *status* to RECEIVED. This will ensure that messages are processed.

BASIC PROCEDURES

This section describes how to access the software, use basic commands, and end a session.

The Message Subsystem does not have a basic operator interface. Basic procedures for operators consist of starting up and shutting down the Message Subsystem. Users of the Message Subsystem interface through email. The software requires minimal operator interface through manipulating database records and the direct processing of messages (see [Advanced Procedures](#)).

Obtaining Help

To obtain help for using the Message Subsystem programs and parameters refer to the manual pages for the components of the subsystem and the software design description of the *Message Subsystem* [\[IDC7.4.2\]](#).

ADVANCED PROCEDURES

This section provides detailed instructions for using the software's advanced features.

▼ Operational Procedures

Changing Status in msgdisc Table

Message Subsystem operators may use the ORACLE database and SQL*Plus to manipulate the **msgdisc** table's *status* field. Message Subsystem programs query the **msgdisc** table for messages to process based on the state listed in the *status* field. Message states and their definitions are given in [Table 3](#).

TABLE 3: MESSAGE PROCESSING STATES

msgdisc.status	Definition
RECEIVED	Message was received by <i>MessageReceive</i> .
QUEUED	Message was queued by <i>MessageGet</i> .
RUNNING	Message is being processed.
DONE	Message processing has completed.
FAILED	Message processing failed, or no data was available to fulfill the request.
NEW	Message is an outbound message.
DONE-PARTIAL	Message processing has completed for some data types in message, and others parts failed or had no available data to fulfill the request.
KILLED	Message processing halted.
STANDBY	Message processing encountered hardware problem while accessing wfdisc files.

Operators can control the processing of messages by changing the message *status* to any of the states listed in [Table 3](#) before the message reaches *status* = **RUNNING**. Operators can block the processing of a message by changing its *status* to an undefined state. Undefined states are ignored by Message Subsystem programs.

The *ParseData* and *AutoDRM* queues can be blocked for each *msgtype* and *msgsub-type* pair as follows: Update the *status* of a number of messages equal to the parameter-controlled² maximum number of processes to *status* = **RUNNING**. Afterwards, *MessageGet* will not invoke another instance, effectively blocking process-

ing. For example, in the event of a failure of a mass storage device, *AutoDRM* can be prevented from attempting to access the offline hardware by blocking the queue for waveform requests as follows:

1. Query the **msgdisc** table for rows with *msgtype* REQUEST with the desired *msgsubtype*, and note the *msgid* numbers for the number of processes needed.
2. Update the rows in the **msgdisc** table for each *msgid* number.

```
SQL> update msgdisc set status='RUNNING'
      where msgid=(insert msgid values here);
```

Processing Messages without Mailer

Messages can be processed directly without use of a mailer by copying the message file with proper email headers directly into the TSD for the Message Subsystem. (Messages require proper email headers to be processed.)

Removing Problem Messages

To remove problem messages that hang the subsystem, remove the message file from the TSD. Messages that have already been processed by *MessageReceive* and are no longer in the TSD can be removed by changing the *status* in the **msgdisc** table or by deleting their records from the database (corresponding entries will be found in the **msgaux** and **msgdatatype** tables).

Software Configuration

Parameter configuration for the Message Subsystem is generally only performed during the initial installation of the software. When the directory structure, database account names, passwords, or machines are changed, pars must be modified.

-
2. The number of allowable processes for each type is specified in the *MessageGet* par file.

▼ Operational Procedures

If changes are made to the par file of any Message Subsystem program except for *AutoDRM* or *ParseData*, the application will need to be stopped and restarted to source the new par values. The Message Subsystem does not have to be stopped and restarted for an *AutoDRM* or *ParseData* par change because these programs source their par files each time they are invoked.

Refer to [“Configuration Data Files” on page 40](#) for information about parameters that need modification for the IDC environment.

MAINTENANCE

Minimal routine maintenance is required for the Message Subsystem. The subsystem generates log files, database entries, and message files. Occasionally, the status of messages in the database tables should be checked for any anomalies.

Log files for the Message Subsystem recycle after a parameter-specified number of instances. No operator maintenance is necessary. A description of how the log files cycle is given in [“Chapter 3: Troubleshooting” on page 27](#).

The size of the database is controlled by data migration and purge of database tables. This maintenance is performed by a separate subsystem, and no maintenance is required by Message Subsystem operators.

Each incoming and outgoing message is stored as a file in the directory structure. These message files require manual archiving by operators. Message files from the directory structure should not be archived to tape until the record from the **msgdisc** table is archived. As an example, the PIDC has configured 42 days of data available in operations. Any older records are moved to the archive database, and their corresponding message files in the directory structure are moved to tape. The PIDC archives these message files approximately once a month.

Data messages available for users via FTP are stored in a directory on a machine outside the firewall. This directory is maintained using *cron* jobs to remove data messages after a time specified in the *cron* job. At the PIDC, *cron* jobs are set to remove these data messages after 30 days.

The *status* field of the **msgdisc** table should be checked periodically. Messages left with *status* = RUNNING for longer than a day may be stalled. Reset *status* to RECEIVED to try processing again. Periodically check for **msgdisc** records with *status* = STANDBY. STANDBY status is assigned when *AutoDRM* is unable to open a file on the archive hardware. Update the *status* to RECEIVED to have the message reprocessed. Records that have *status* = KILLED are not being processed; this status is given when the Message Subsystem is shut down by killing the active subsystem processes. These records should be updated to *status* = RECEIVED.

SECURITY

Security for the Message Subsystem is provided by ownership of the processes. Operators will be able to start and terminate the Message Subsystem if they are working with the same UNIX user login that has ownership of the processes.

Forged messages can be sent by unauthorized external users to the subsystem for processing. These messages could contain false data to be parsed into the database accounts or request data from the database. Future releases address this risk by authenticating messages.

If operators have update permissions for the database accounts, they can potentially remove, add, or manipulate data in the account. For example, messages from a particular nation or site can be blocked from processing by changing the *status* column of the record in the **msgdisc** table from RECEIVED to some other state. Tables can be dropped, deleted, or truncated. Records from the tables can be removed or updated.

Authorized external users of the Message Subsystem are tracked by the **datauser** table. Each user is identified by a unique user name and domain, which must match all email headers of messages sent to the subsystem to be processed. Users are separated into different classes to prioritize processing of messages.

Passwords

Passwords are not written to Message Subsystem log files.

▼ Operational Procedures

Database passwords and accounts are stored in `process.par`. Anyone with proper permission to view this file can retrieve database passwords and has the ability to corrupt database accounts (for example, truncate a table, drop a table, remove `wfdisc` records, and so on). Access to these files is controlled using UNIX permissions, and the operations manager controls the UNIX group membership required to access the file.

Passwords to database accounts may be disclosed if a Message Subsystem program exits and generates a segmentation fault or core dump. To prevent this output, a parameter in `global.env` limits the size of the core file generated.

Marking/Storing Controlled Outputs

All incoming and outgoing messages are stored as files in the directory structure. Permissions on these files allow all operators and staff at the IDC to access these messages.

Because email is used as the primary method of transmitting messages, content of the messages is not secure during routing from the originator to the recipient. Email can be intercepted during transmission.

Data compiled per request and made available by FTP is stored as a file in a directory structure. An email notifies the requestor that the data is available for retrieval via FTP. Data files stored in this directory for retrieval are accessible by all users.

Chapter 3: Troubleshooting

This chapter describes how to identify and correct problems related to the Message Subsystem and includes the following topics:

- [Monitoring](#)
- [Interpreting Error Messages](#)
- [Solving Common Problems](#)
- [Reporting Problems](#)


```
% ls -l
total 96431
-rw-rw-r-- 1 testbed testbed 467 Apr 11 2000 AutoDRM
-rw-rw-r-- 1 testbed testbed 416 Apr 01 2000 AutoDRM.1
-rw-r--r-- 1 testbed testbed 342 Mar 20 2000 AutoDRM.10
-rw-r--r-- 1 testbed testbed 342 Mar 20 2000 AutoDRM.11
-rw-r--r-- 1 testbed testbed 342 Mar 20 2000 AutoDRM.12
-rw-r--r-- 1 testbed testbed 342 Mar 20 2000 AutoDRM.13
-rw-rw-r-- 1 te ...
```

AutoDRM and *ParseData* create a new log file each time they are invoked (for each message sent to the subsystem). As a result, these log files cycle more quickly than the other Message Subsystem programs. The other subsystem programs, such as *MessageReceive*, create a log file when the Message Subsystem is invoked and continually append to their respective log file.

Screening TSD

The TSD can be monitored to verify that messages are being processed. After a message is sent to the Message Subsystem, observe the entry into the TSD. Messages are moved out of the TSD as they enter into the tracking system, and any messages remaining in the TSD may indicate an error.

```
% cd msg2_TSD
% ls -la
total 2
-rw-r--r-- 1 nmrd testbed 875 Jun 26 1999 msg.AAA0lp0Y
-rw-r--r-- 1 nmrd testbed 876 Jun 26 1999 msg.AAAvxxkau
```

The Message Subsystem should process messages within 1–2 minutes. Check the directory to verify messages have been processed.

```
% ls -la
total 0
```

▼ Troubleshooting

Verifying Auxiliary Data Processed

Auxiliary data processing is a function specific to the Message Subsystem. An absence of auxiliary data may indicate an error with *ParseData* or *MessageGet*. To ensure that auxiliary data are processed, query the **request** table for the most recent record with *state* = done-success. This will show how recently auxiliary data was processed. For example:

```
SQL> select max(start_time) from idcx.request
      where state='done-success';
```

The *RequestFlow* GUI tool normally is used to verify that auxiliary data are being processed. *RequestFlow* monitors the **request** table and graphically indicates the most recent auxiliary data received.

To verify that auxiliary data are being requested, track the *reqid* in the **request** table to the *intid* in the **msgdisc** table to the *msgid* in the **msgdest** table. If the **msgdest** row has *status* = DONE, a request for auxiliary data was emailed. For example, to determine the status of a specific *reqid* use the following query:

```
SQL> select md.status
      from idcx.request r, idcx.msgdisc m, idcx.msgdest md
      where r.reqid=reqid
            and m.intid=r.reqid
            and m.intidtype='reqid'
            and md.msgid=m.msgid;
```

Monitoring Status in Database Tables

The **msgdisc**, **msgdest**, and **msgaux** database tables can be queried to verify records are being processed. Check the processing of a message by querying the **msgdisc** table with the *msgid*, and verify that *status* is updated. An error may have occurred if a message remains in any state for an extended length of time (for example, one day).

```
SQL> select msgid, itime, idate, status from msgdisc
      where msgid=5947527;
```

MSGID	ITIME	IDATE	STATUS
5947527	940464219	1999294	RUNNING

In this example, the msgdisc record with *msgid* = 5947527 has *status* = RUNNING. After a moment, repeat the same query to verify the message is being processed. The *status* field should be updated to another status.

Screening Log Files

The log files of the individual programs can be screened for “error” or “fatal” messages. To do so, change directories to the Message Subsystem log file directory. The following list displays a sample log directory:

```
% cd /data/peas/misc/log/msg/
% ls -l
total 2578
-rw-rw-r-- 1 zatlouka testbed      443 Dec  9 1997 AutoDRM
-rw-rw-r-- 1 zatlouka testbed      443 Dec  9 1997 AutoDRM.1
-rw-rw-r-- 1 zatlouka testbed      585 Dec  9 1997 AutoDRM.10
-rw-rw-r-- 1 zatlouka testbed      585 Dec  9 1997 AutoDRM.11
-rw-rw-r-- 1 zatlouka testbed      585 Dec  9 1997 AutoDRM.12
-rw-rw-r-- 1 zatlouka testbed      585 Dec  9 1997 AutoDRM.13
-rw-rw-r-- 1 zatlouka testbed      585 Dec  9 1997 AutoDRM.14
-rw-rw-r-- 1 zatlouka testbed      443 Dec  9 1997 AutoDRM.15
-rw-rw-r-- 1 zatlouka testbed      585 Dec  9 1997 AutoDRM.16
-rw-rw-r-- 1 zatlouka testbed      443 Dec  9 1997 AutoDRM.17
-rw-rw-r-- 1 za ...
```

The current log file for any program is the program name. To look at a program's previous iteration log file, examine *progrname.1*. If the Message Subsystem is stopped and restarted, *progrname.1* becomes *progrname.2* and continues this numbering cycle until log files recycle.

Use the UNIX `grep` command to search for problems:

```
% grep fatal AutoDRM
% grep fatal AutoDRM.*
% grep error AutoDRM
% grep error AutoDRM.*
```

▼ Troubleshooting

```
AutoDRM.10:/home/zatlouka/MSG_SYS.GAP/bin/AutoDRM[error]: #612738:
type=no data, text=There are no waveforms in the database which
matched this request.
```

To monitor the operation of any Message Subsystem program (except for *AutoDRM* or *ParseData*) in real-time, use the UNIX `tail -f` command on its log file to monitor the standard output of the program.

INTERPRETING ERROR MESSAGES

Log files should be examined whenever the `msgdisc.status` or `msgaux.substatus` fields indicate an error other than Parse Error or No Data. Parse Error occurs when *ParseData* attempts to process an improperly formatted data message. No Data occurs when no data were available for the user request.

Message: `db connection error - server timed out`

Description: Cannot connect to database account to write to tables or read information.

Action: Attempt to connect to database.

Message: `no data`

Description: Cannot access data.

Action: Try to access database account with connect string. Manually query for requested data. If there are no data, the system functioned properly.

Message: `parse error`

Description: Error parsing data.

Action: None. User induced error within message received.

Message: no data available

Description: No data exist for the user request.

Action: None.

Message: cannot open file/ cannot access file

Description: Program cannot open specified file.

Action: Check file and directory permissions. It is possible that a database entry points to a file that does not exist. It is also possible that the automounter or hardware failed.

SOLVING COMMON PROBLEMS

Message processing can be interrupted without a program writing an error message to a log file. Check for the following common problems if the Message Subsystem appears to have stopped processing.

Message processing may have been halted due to an unsigaled exit of a program. Verify that an *AutoDRM* or a *ParseData* process is running for the message(s) sent to the subsystem. Verify that *MessageGet*, *MessageShip*, and *MessageReceive* processes are running using the command `ps -ef | grep Message`. If *MessageGet*, *MessageShip*, or *MessageReceive* have exited, refer to ["Software Startup" on page 18](#) for instructions on restarting them. If the *keep_msg_alive* script is used, these programs will be restarted without operator assistance. Verify processes for other Message Subsystem programs are active by examining the date/time of the last activity on the log files.

Message processing may have been halted due to either *AutoDRM* or *ParseData* reaching their active process limit. These two programs have a par-settable number of active processes allowed, which are set in the *MessageGet* par file. If the *MessageGet* par file specifies a maximum of eight *AutoDRM* processes, and eight *AutoDRM* processes are running, *MessageGet* will not invoke another *AutoDRM* process. Verify that the specified number of processes are changing process ID

▼ Troubleshooting

numbers. If any process ID numbers are displayed for an abnormal length of time (for example, one day), they are probably stale processes that have hung up the subsystem. Kill the stale processes. Update the **msgdisc** record(s) with *status* = **RUNNING** to *status* = **RECEIVED**. Messages in *status* = **RUNNING** are not processed when the subsystem is restarted and can block the processing queue.

Error Recovery

If an error occurs with the Message Subsystem, there are several ways to identify the cause of the problem and return the subsystem back to normal operations.

The Message Subsystem distributes email about errors to the “operator” email address(es) specified in the **shared.par** file. This mail is usually distributed when a problem or unknown message type is sent to the Message Subsystem. The problem message is forwarded to the operators for analysis.

To discover the cause of the error directly, examine the log files of the Message Subsystem programs. Error and other valuable debugging information is written to the log files during the program’s operation. These errors would include the typically encountered errors described in the previous section.

Another way to determine the cause of or to retrieve more information about an error is to query the **msgaux** table for *substatus*. The *substatus* field shows errors encountered during message processing including the probable error, the line number in the code the error was encountered, the command executed, and the error generated to the log file. This information is useful for debugging purposes.

If a specific message causes *AutoDRM* to exit, the message can be retrieved for debugging purposes. *Msgids* are written to the *AutoDRM* log file as messages are processed. To retrieve the message, get the *msgid* from the *AutoDRM* log file and query the **msgdisc** table for the record with the *msgid*. Then, retrieve the message from the message directory, and examine it for errors.

These methods will help troubleshoot problems that may occur. After a problem has been identified, check the status of the Message Subsystem. If *MessageGet*, *MessageShip*, or *MessageReceive* has exited, restart them. Refer to [“Software Startup” on page 18](#) for instructions. If the *keep_msg_alive* script is used, these programs will be restarted without operator assistance.

REPORTING PROBLEMS

The IDC will use the following procedures for diagnosing and reporting software defects while PIDC staff are on-site at the IDC. After PIDC staff are no longer on-site, or in other situations, a different model will apply.

1. The IDC will first analyze the fault to confirm that it is associated with the software rather than with IDC infrastructure or operator error. A “fault” is any problem encountered in operating this software at the IDC. A “defect” is a fault caused by a software bug or other problem associated with the software and its configuration.
2. The IDC will use a system to record, store, and distribute messages related to faults. An accurate record of faults encountered during IDC operations is needed for evaluating the IDC performance against reliability requirements and for deploying resources to correct recurrent or significant faults.
3. The IDC will classify each fault according to its most likely type (for example, hardware infrastructure, software infrastructure, application subsystems, and so on). Initial diagnosis of the fault will require use of an operating system and database tools, scripts to automate use of these tools, and log files written by COTS and application software.
4. The IDC will describe the fault symptoms and the results of the initial diagnosis, as necessary, in the initial log entry. If a problem has been resolved prior to the log entry, they will also describe the solution. If the technical problem remains unresolved, they will log additional entries as supplementary diagnostic information becomes available. PIDC staff in Vienna will assist with the initial diagnosis.

▼ Troubleshooting

5. Upon conclusion that the fault is a software or system-configuration defect, the IDC will report this defect and the results of initial diagnosis to PIDC staff in Vienna. A “defect-report form” has been provided for this purpose, and all defect reports will be logged and tracked until final action has been taken to resolve them.
6. Upon receiving a defect report, PIDC staff will confirm the IDC classification as a defect and will categorize this defect and conduct further diagnosis. Each defect will be categorized according to its effect on executing the operational concept.
7. PIDC staff will acknowledge receipt of the defect report and assign the defect to a severity category. As necessary, Vienna-based PIDC staff will consult informally with U.S.-based PIDC staff to complete the defect isolation configuration mistakes at the IDC. They will also flag similar defects that may have been reported from other sites and review source code to interpret error messages from the log files.
8. After diagnosis is complete, PIDC staff will initiate the actions required to clear the defect. In some cases a straightforward change of configuration will correct the defect. In other cases a change to the application software will be required, and Vienna-based PIDC staff will submit a software modification request (SMR) to the PIDC development group. The SMR will include the defect categorization and a summary of the diagnostic results.
9. The PIDC staff will inform the IDC on the issuance of the SMR and keep IDC staff informed on the process of problem resolution at the PIDC. The SMRs will be stored within the IDC system.

Chapter 4: Installation Procedures

This chapter provides instructions for installing the software and includes the following topics:

- [Preparation](#)
- [Executable Files](#)
- [Configuration Data Files](#)
- [Database](#)
- [Initiating Operations](#)
- [Validating Installation](#)

Chapter 4: Installation Procedures

PREPARATION

Determine which machine will be used as the Message Subsystem host. A directory structure must be created to support the Message Subsystem. Paths to these directories will need to be modified in the par files as discussed in “[Configuration Data Files](#).”

msg_TSD/	temporary storage directory for incoming messages
msgftp_TSD/	temporary storage directory for <i>MessageFTP</i>
aux_TSD/	temporary storage directory for auxiliary data
log/msg/	directory for Message Subsystem program log files
msg/	archive directory for incoming and outgoing messages
msg/staging/	working directory for <i>MessageReceive</i>
invalidmail/	directory for invalid mail

The applications *MessageStore* and *MessageReceive* should run on the host where the Message Subsystem directories reside. The use of NFS can hinder performance of the Message Subsystem.

Obtaining Released Software

The software is obtained via FTP from a remote site or via a physical medium, such as tape or CD-ROM. The software and associated configuration data files are stored as one or more tar files. The software and data files are first transferred via FTP or copied from the physical medium to an appropriate location on a local hard disk. The tar files are then untarred into a standard UNIX directory structure.

Hardware Mapping

The user must select the hardware on which to run the software components. Software components are generally mapped to hardware to be roughly consistent with the software configuration model.

UNIX System

The operations system administrator will need to create three mail aliases for the Message Subsystem. One alias is used for *AutoDRM*. A second alias receives incoming message mail. A third alias directs incoming messages to the Message Subsystem host machine, ensuring that *MessageStore* will run on the mail host.

Messages come into the Message Subsystem through a series of mail aliases with the final step invoking a process directly out of the mail system using sendmail pipes.

The *MISCHOST* value is symbolic for the hostname of the miscellaneous and auxiliary processing server. During the installation process the actual hostname must be used in the alias. The file */LOCALPATH/run_MessageStore* must be owned by the auto user and stored directly on the system disk of *MISCHOST*. The *LOCAL-PATH* value is symbolic for the path to a directory specifically designated for the file *run_MessageStore*.

During installation the *run_messageStore* file will be properly configured to insert the data into the Message Subsystem.

▼ Installation Procedures

TABLE 4: MESSAGE SUBSYSTEM MAIL ALIASES

Name	Purpose
autodrm	messages
messages	<i>MISCHOST-MESSAGES@MISCHOST</i> receives incoming mail
<i>MISCHOST-MESSAGES@MISCHOST</i>	:include:/LOCALPATH/ run_messageStore directs incoming messages to the Message Subsystem host machine

Firewall

The system administrator must configure the firewall at the IDC to allow FTP and email processes for the Message Subsystem. The firewall must be configured to allow incoming email and to allow outgoing FTP for the retrieval of auxiliary waveform data. The firewall must also be configured to support the use of FTP to place data messages outside the firewall for pickup by outside users.

EXECUTABLE FILES

Message Subsystem executable files should be installed in the `/cmss/rel/bin/` directory accessible by the Message Subsystem host machine. Scripts should be installed in the `/cmss/scripts/` directory. Refer to [Table 2 on page 11](#) for a list of executables.

CONFIGURATION DATA FILES

Pars have default values in place except where paths are set. For efficiency, Message Subsystem par files use global variables for path names where possible.

The files `process.par`, `shared.par`, and `msgs.par` in the subdirectory `/cmss/config/system_specs/` contain IDC specific paths and aliases that need modification for the IDC environment. The file `global.shenv` located at `/cmss/`

config/system_specs/env/ also contains global environment variables that need modification for the IDC environment. These files contain global variables for all operational software. The following examples of these par files show only Message Subsystem applicable entries in these files:

process.par (IMSPAR)

```
IDCXDB=account/password@machine
IDCX-ARCHDB=account/password@machine
EXPERTDB=$(IDCXDB)
par=$(CMS_CONFIG)/system_specs/shared.par
```

shared.par

```
operator=name, name, name
NOTIFY_LIST=name, name, name
domain=abc.def
DATABASE_VENDOR=oracle

# Shared Directories
LOGDIR=message_path/misc/log
AUXDIR=message_path/w
MSGDIR=message_path/msg

# Shared Directories with derived paths
RELDIR=$(CMS_HOME)
RELBIN=$(RELDIR)/bin
STATICDIR=$(CMS_CONFIG)/earth_specs
SQLDIR=$(CMS_CONFIG)/system_specs/sql
GLOBALDIR=$(CMS_CONFIG)/system_specs
CONTRIB_BIN=$(CONTRIB_HOME)/bin
SCRIPTSBIN=$(CMS_SCRIPTS)/bin

# Each subsystem has a fixed path to it.
MESSAGES-DIR=$(CMS_CONFIG)/app_config/messages

# The following define the subsystem specific par files
MESSAGES=$(CMS_CONFIG)/system_specs/msgs.par

# machines
MSGHOST=machinename
```

▼ Installation Procedures

msgs.par

```

PARDIR=$(MESSAGES-DIR)

# Common log directory for all message system applications
MSGLOGDIR=$(LOGDIR)/msg

# Common message staging directory
MSG_TSD=message_path/misc/msg_TSD

# message return address
MSG_RETURN_ADDRESS=user@address

# Mail addresses to receive notifications from ParseData that
# a message failed for some reason.
MSG_OPERATOR="name,name,name"

# Name of the RMS pipeline command:
RMSPipeline=/home/rmsops/rms_pipeline

```

global.env

```

APPDIR=/var/tuxedo/PIDC62_process:/cmss/rel/bin:/home/oracle/lib
ATRIAHOME=/opt/atria
CMS_APPS=/cmss/config/app_config
CMS_CONFIG=/cmss/config
CMS_HOME=/cmss/rel
CMS_MODE=process
CMS_RELEASE=PIDC62
CMS_SCRIPTS=/cmss/scripts
CONTRIB_HOME=/cmss/contrib
GDIHOME=/cmss/rel
GDI_HOME=/cmss/rel

GS_LIB=/cmss/local/lib/ghostscript/3.33:
        /cmss/local/lib/ghostscript/fonts

HOME=/home/pipeline
IMSPAR=/cmss/config/system_specs/process.par
IPC_GROUP=DACS

LD_LIBRARY_PATH=/usr/dt/lib:/cmss/rel/lib:/home/oracle/lib:
        /cmss/cots/tuxedo/lib:/opt/SUNWsprow/lib:/opt/SUNWmotif/lib:
        /usr/openwin/lib:/cmss/contrib/lib

LOCALHOME=/cmss/local

MANPATH=/usr/man:/usr/openwin/man:/usr/dt/man:/cmss/rel/doc/man:
        /cmss/local/man

```

```

MOTIFHOME=/opt/SUNWmotif
MOZILLA_HOME=/opt/local/bin
OPENWINHOME=/usr/openwin
ORACLE_HOME=/home/oracle
ORACLE_SID=oracle

PATH=/usr/bin:/usr/sbin:/usr/dt/bin:/cmss/scripts/bin:/cmss/rel/bin:
/cmss/contrib/bin:/home/oracle/bin:/cmss/cots/tuxedo/bin:
/cmss/local/bin:/opt/SUNWspro/bin:/opt/SUNWmotif/bin:
/usr/openwin/bin:/home/licensed/frame_5.5/bin:/opt/atria/bin:
/opt/local/bin:/opt/local/adobe/Acrobat3/bin:/etc:/usr/ccs/bin:
/usr/ucb:.

PERLLIB=/cmss/scripts/lib
PERLPATH=/cmss/local/bin/
SCHEMEMPATh=/cmss/rel/scheme
SPROHOME=/opt/SUNWspro
TLOGPFX=/var/tuxedo/PIDC62_process/TLOGS/tlog

```

DATABASE

This section describes database elements, including accounts, tables, and initialization of the **lastid** table, required for operation of the Message Subsystem.

Accounts

AutoDRM needs to access both the operations and archive databases typically through the primary pipeline account (IDCX). In addition, *AutoDRM* queries other accounts to retrieve data (SEL1, SEL2, SEL3, REB, and so on). The tables in these accounts queried by *AutoDRM* must be able to read the tables in these accounts. Finally, *AutoDRM* must be able to write to the **msgdisc**, **msgdest**, **msgaux** and **msgdatatype** tables, which are typically in the IDCX account.

Tables

The database account for the *Message Subsystem* must contain the tables that are part of the *Message Subsystem* functional group (see [\[IDC5.1.3\]](#)). [Table 5](#) lists these tables. [Figure 5](#) shows the entity relationships between the major tables of the *Message Subsystem*.

▼ Installation Procedures

TABLE 5: MESSAGE SUBSYSTEM DATABASE TABLES

Table	Description
datauser	This table tracks authorized users of the <i>Message Subsystem</i> . Each user is identified by a user name and domain, which must match all email headers.
ftpfailed	This table facilitates FTP retrieval and placement of data messages between the IDC and contributing National Data Centers (NDCs).
ftplogin	This table contains logon information for FTP retrieval. The rows are used by the auxiliary data retrieval system to obtain data via FTP from auxiliary stations. This table should not be publicly readable.
msgaux	This table records the cause of failed messages.
msgdatatype	This table supports data tracking by recording each data section in a message for both incoming and outgoing data messages.
msgdest	This table contains information about the destination of messages sent from the IDC.
msgdisc	This table contains information pertinent for messages including the date and time the message was sent or received, identification information, and where the message is stored.
outage	This table contains information on the availability of waveform data.
xtag	This table links various identifiers, for example, <i>orid</i> , <i>stassid</i> , and <i>wfid</i> to other identifiers.

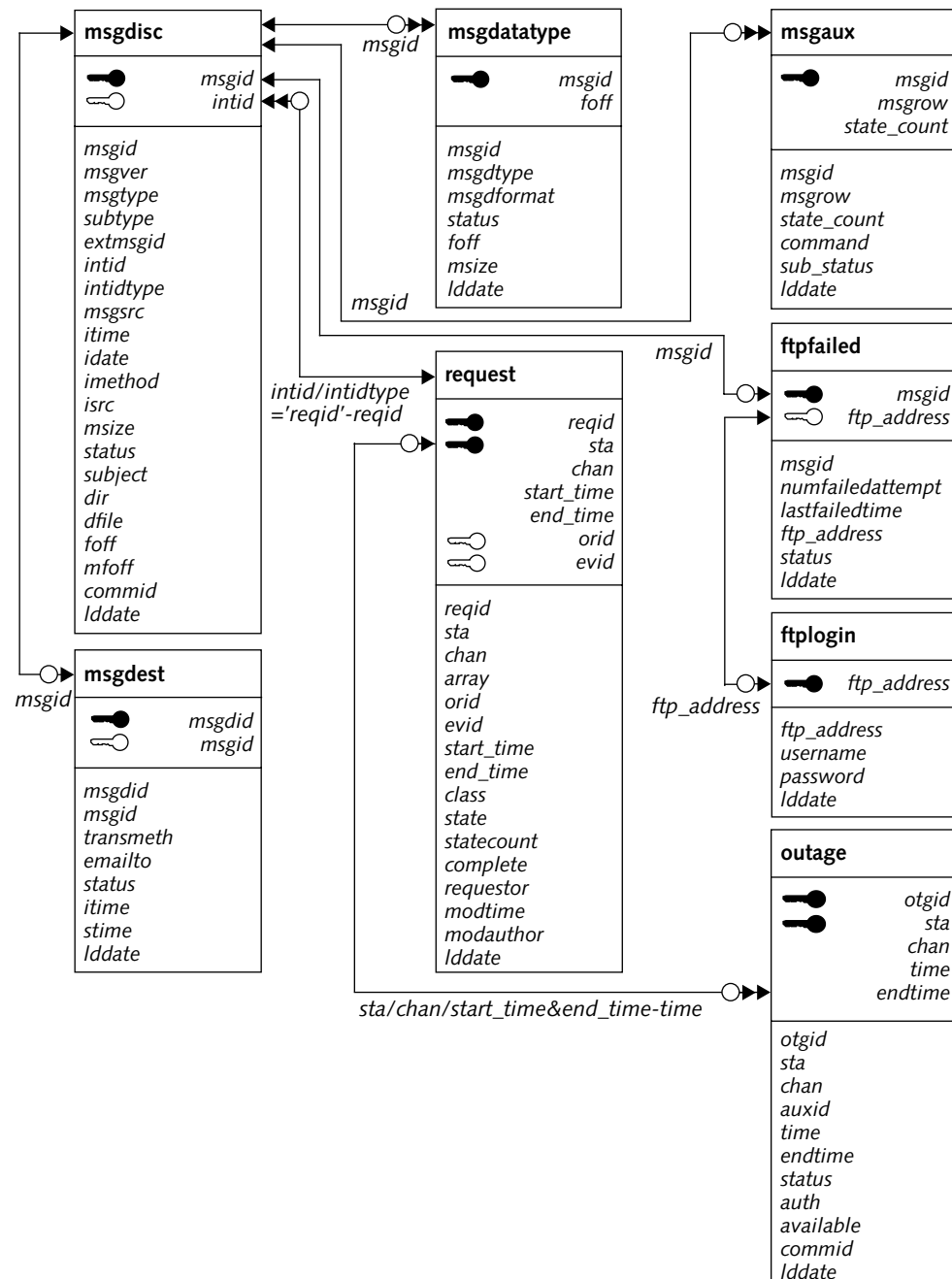


FIGURE 5. ENTITY RELATIONSHIPS OF MESSAGE SUBSYSTEM TABLES

▼ Installation Procedures

Attributes, variable types and length, and other information needed to create *Message Subsystem* tables can be found in [\[IDC5.1.1Rev2\]](#) and [\[IDC5.1.3\]](#). SQL *Plus scripts containing this information are used to create the database tables listed in [Table 5](#). For example, the SQL *Plus script used to create the **msgdisc** table follows:

msgdisc.sql

```

create table MSGDISC (
msgid      NUMBER(8)      NOT NULL,
msgver     VARCHAR2(8),
msgtype    VARCHAR2(16),
subtype    VARCHAR2(2),
extmsgid    VARCHAR2(20),
intid       NUMBER(8),
intidtype   VARCHAR2(16),
msgsrc     VARCHAR2(16),
itime       FLOAT(53),
idate       NUMBER(8),
imethod     VARCHAR2(8),
isrc        VARCHAR2(64),
msize       NUMBER(8),
status      VARCHAR2(32),
subject     VARCHAR2(64),
dir         VARCHAR2(64),
dfile       VARCHAR2(32),
foff        NUMBER(8),
mfoff       NUMBER(8),
commid      NUMBER(8),
lddate      DATE
) tablespace IDC storage ( initial 5m next 1m ) pctfree=10;

create unique index MSGIDNX on MSGDISC(MSGID)
tablespace IDCNDX storage ( initial 1m next 1m ) pctfree=10;

create index MSGIDATENX on MSGDISC(IDATE)
tablespace IDCNDX storage ( initial 1m next 1m ) pctfree=10;

create index INTIDNX on MSGDISC(INTID)
tablespace IDCNDX storage ( initial 1m next 1m ) pctfree=10;

create index MSGSTATNX on MSGDISC(STATUS)
tablespace IDCNDX storage ( initial 1m next 1m ) pctfree=10;

```

```
grant SELECT on MSGDISC to PUBLIC ;
grant DELETE on MSGDISC to MIGRATE ;
```

Initialization of LastID

The *Message Subsystem* requires five **lastid** table attributes for operation. These attributes must be added to the **lastid** table (if the attributes are not already there) and must be initialized to non-negative values (0 is acceptable). An example of the *Message Subsystem* attributes in the PIDC **lastid** table follows:

KEYNAME	KEYVALUE	LDDATE
-----	-----	-----
dfid	99495	13-SEP-1999
msgid	5933373	13-OCT-1999
msgdid	3130720	13-OCT-1999
userid	1370	11-OCT-1999
fpid	59193	13-OCT-1999

INITIATING OPERATIONS

The Message Subsystem is ready for startup after new directories are made, executables are installed, the new accounts and mail aliases are established, new database accounts are created and populated with the required tables, the **lastid** table is initialized, and parameters in the par files are modified for the new environment. Refer to [“Software Startup” on page 18](#) for instructions on starting the Message Subsystem.

VALIDATING INSTALLATION

To validate the installation of the Message Subsystem, send messages to and receive messages from the Message Subsystem. Example messages follow:

A request message:

```
BEGIN IMS1.0
MSG_TYPE REQUEST
MSG_ID 1111
```

▼ Installation Procedures

```
E-MAIL add@email.here  
TIME 1999/10/17 23:44:00 TO 1999/10/18 00:30:00  
BULL_TYPE REB  
ORIGIN IMS1.0  
STOP
```

A help message:

```
BEGIN IMS1.0  
MSG_TYPE REQUEST  
MSG_ID ANY_NDC  
E-MAIL add@email.here  
HELP  
STOP
```

For further methods of verifying installation, refer to ["Monitoring" on page 28](#).

References

The following sources supplement or are referenced in the document:

- [And90a] Anderson, J., Farrell, W., Garcia, K., Given, J., and Swanger, H., *CSS Version 3 Database: Schema Reference Manual*, Science Applications International Corporation, 1990.
- [IDC3.4.1Rev2] Science Applications International Corporation, Veridian Pacific-Sierra Research Corporation, *Formats and Protocols for Messages, Revision 2*, SAIC-00/3005, PSR-00/TN2829, 2000.
- [IDC3.4.3] Science Applications International Corporation, *Formats and Protocols for Continuous Data, CD-1.1*, SAIC-00/3026, 2000.
- [IDC5.1.1Rev2] Science Applications International Corporation, Veridian Pacific-Sierra Research Corporation, *Database Schema, Revision 2*, SAIC-00/3057, PSR-00/TN2830, 2000.
- [IDC5.1.3] Science Applications International Corporation, Pacific-Sierra Research Corporation, *Configuration of PIDC Databases*, SAIC-99/3019, PSR-99/TN1114, 1999.
- [IDC7.4.2] Science Applications International Corporation, Pacific-Sierra Research Corporation, *Message Subsystem*, SAIC-98/3003, 1998.
- [WGB97] Working Group B, *Operational Manual for the International Data Centre*, Preparatory Commission of the Comprehensive Nuclear Test-Ban Treaty Organization, WGB/TL/44, 1997.

Glossary

A

alias

(1) Name assigned to a command sequence. (2) Name encompassing several user names.

ASCII

American Standard Code for Information Interchange. Standard, unformatted 256-character set of letters and numbers.

attribute

A database column.

authentication signature

Series of bytes that are unique to a set of data and that are used to verify the authenticity of the data.

AutoDRM

Automatic Data Request Manager.

automounter

System daemon on a workstation that mounts partitions from remote file servers on demand.

C

CD-ROM

Compact Disk–Read Only Memory.

child process

UNIX process created by the *fork* routine. The child process is a snapshot of the parent at the time it called *fork*.

command

Expression that can be input to a computer system to initiate an action or affect the execution of a computer program.

Computer Software Component

Functionally or logically distinct part of a computer software configuration item, typically an aggregate of two or more software units.

Computer Software Configuration Item

Aggregation of software that is designated for configuration management and treated as a single entity in the configuration management process.

configuration

Arrangement of computer system or component as defined by the number, nature, and interconnection of its parts.

▼ Glossary

COTS

Commercial-Off-the-Shelf; terminology that designates products such as hardware or software that can be acquired from existing inventory and used without modification.

CSC

Computer Software Component.

CSCI

Computer Software Configuration Item.

E**email**

Electronic mail.

execute

Carry out an instruction, process, or computer program.

F**fork**

Create a child process.

FTP

File Transfer Protocol; protocol for transferring files between computers.

G**GB**

Gigabyte. A measure of computer memory or disk space that is equal to 1,024 megabytes.

GSE

Group of Scientific Experts.

GUI

Graphical User Interface.

H**host**

Machine on a network that provides a service or information to other computers. Every networked computer has a hostname by which it is known on the network.

I**ID**

Identification; identifier.

IDC

International Data Centre.

IDC Operators

Technical staff that install, operate, and maintain the IDC systems and provide additional technical services to the individual States Parties.

IMS

International Monitoring System.

K**KB**

Kilobyte. 1,024 bytes.

M

MB

Megabyte. 1,024 kilobytes.

message type

Kind of message; possible message types include DATA, REQUEST, and SUBSCRIPTION.

message ID

Unique 20-character alphanumeric identification given to a message by the sender that facilitates message tracking for the sender.

N

NDC

National Data Center.

NFS

Network File System (Sun Microsystems). Protocol that enables clients to mount remote directories onto their own local filesystems.

O

ORACLE

Vendor of PIDC and IDC database management system.

orid

Origin Identifier.

origin

Hypothesized time and location of a seismic, hydroacoustic, or infrasonic event. Any event may have many origins. Characteristics such as magnitudes and error estimates may be associated with an origin.

P

parameter (par) file

ASCII file containing values for parameters of a program. Par files are used to replace command line arguments. The files are formatted as a list of [*token* = *value*] strings.

PIDC

Prototype International Data Centre.

R

radionuclide

REB

Reviewed Event Bulletin; the bulletin formed of all S/H/I events that have passed analyst inspection and quality assurance review.

S

schema

Database structure description.

▼ Glossary

SEL1

Standard Event List 1; S/H/I bulletin created by total automatic analysis of continuous timeseries data. Typically, the list runs one hour behind real time.

SEL2

Standard Event List 2; S/H/I bulletin created by totally automatic analysis of both continuous data and segments of data specifically down-loaded from stations of the auxiliary seismic network. Typically, the list runs five hours behind real time.

SEL3

Standard Event List 3; S/H/I bulletin created by totally automatic analysis of both continuous data and segments of data specifically down-loaded from stations of the auxiliary seismic network. Typically, the list runs 12 hours behind real time.

S/H/I

Seismic, hydroacoustic, and infrasonic.

SMR

Software Modification Request.

SMTP

Simple Mail Transfer Protocol.

software unit

Discrete set of software statements that implements a function; usually a sub-component of a CSC.

Solaris

Name of the operating system used on Sun Microsystems hardware.

SQL

Structured Query Language; a language for manipulating data in a relational database.

States Parties

Treaty user group who will operate their own or cooperative facilities, which may be NDCs.

subsystem

Secondary or subordinate system within the larger system.

T

tar

Tape archive. UNIX command for storing or retrieving files and directories. Also used to describe the file or tape that contains the archived information.

TCP/IP

Transmission Control Protocol/Internet Protocol.

U

UNIX

Trade name of the operating system used by the Sun workstations.

W

waveform

Time-domain signal data from a sensor (the voltage output) where the voltage has been converted to a digital count

(which is monotonic with the amplitude of the stimulus to which the sensor responds).

Web

World Wide Web; a graphics-intensive environment running on top of the Internet.

wfdisc

Waveform description record or table.

workstation

High-end, powerful desktop computer preferred for graphics and usually networked.

Index

A

active process limit [33](#)
archiving [24](#)
AutoDRM [5](#), [7](#), [10](#), [14](#)
 active process limit [33](#)
 blocking queue [22](#)
 database accounts [43](#)
 error recovery [34](#)
 gradual shutdown [20](#)
 immediate shutdown [20](#)
 log files [29](#)
 mail alias [39](#)
 reconfiguring [24](#)
 standby status [25](#)
 verify that running [33](#)
aux_TSD/ [38](#)
auxiliary data processing verification [30](#)

B

blocking message processing [22](#)

C

cannot access file [33](#)
cannot open file [33](#)
command line [14](#)
common problems [33](#)

compress [14](#)
configuration [40](#)
cron [6](#), [14](#), [18](#), [19](#), [24](#)
crontab [19](#)

D

database size [24](#)
data flow symbols [v](#)
datauser
 description [44](#)
db connection error [32](#)
disk space requirements [14](#)

E

.end [20](#)
entity-relationship diagram [45](#)
entity-relationship symbols [v](#)
environment [40](#)
 software [14](#)
error messages interpretation [32](#)
error recovery [34](#)
executable files location [40](#)

F

firewall [40](#)
FTP directory [24](#)
ftpfailed [45](#)
 description [44](#)
ftlogin [45](#)
 description [44](#)

▼ Index

G

`global.env` [42](#)
 global variables [40](#)
gzip [14](#)

H

hang [vi](#)

I

IDCX database account [43](#)
 installation [38](#)
 instance [vi](#)
 invalidmail/ [38](#)
 inventory [11](#)
 invoke [vi](#)

K

keep_msg_alive [11](#), [14](#), [19](#), [33](#), [35](#)
 KILLED [25](#)

L

lastid [vi](#), [47](#)
 log/msg/ [38](#)
 log files [28](#)
 screening [31](#)

M

mail aliases [39](#)
mailx [6](#), [14](#)
 maintenance [24](#)
 man pages [iii](#)
 memory requirements [14](#)

message
 archiving [24](#)
 inbound flow [8](#)
 outbound flow [9](#)
 storage [26](#)
MessageAlert [5](#), [14](#)
 gradual shutdown [20](#)
 immediate shutdown [20](#)
MessageFTP [5](#), [7](#), [14](#)
 temporary storage directory [38](#)
MessageGet [6](#), [7](#), [22](#), [30](#)
 error recovery [35](#)
 immediate shutdown [20](#)
 startup [18](#)
 verify that running [19](#), [28](#), [33](#)
MessageGet.par [10](#)
MessageReceive [6](#), [7](#), [23](#)
 error recovery [35](#)
 gradual shutdown [20](#)
 immediate shutdown [20](#)
 log files [29](#)
 startup [18](#)
 verify that running [19](#), [28](#), [33](#)
 working directory [38](#)
 messages
 temporary storage directory [38](#)
MessageShip [6](#), [7](#)
 error recovery [35](#)
 gradual shutdown [20](#)
 immediate shutdown [20](#)
 startup [18](#)
 verify that running [19](#), [28](#), [33](#)
MessageStore [6](#)
 and mail alias [39](#)
 host [38](#)
 startup [18](#)

Message Subsystem

- configuration [40](#)
- gradual shutdown [19](#)
- host [38](#)
- immediate shutdown [20](#)
- installation [38](#)
- inventory [11](#)
- log file directory [38](#)
- maintenance [24](#)
- monitoring [28](#)
- passwords [25](#)
- restarting automatically [19](#)
- security [25](#)
- starting [18](#)
- validating installation [47](#)
- verifying active processes [28](#)
- verifying operation [19](#), [28](#)
- MessageGet*
 - gradual shutdown [20](#)
- monitoring [28](#)
- monitoring tools [11](#)
- msg/* [38](#)
- msg/staging/* [38](#)
- msg_TSD/* [38](#)
- msgaux** [34](#), [45](#)
 - description [44](#)
- msgdatatype** [45](#)
 - description [44](#)
- msgdest** [45](#)
 - description [44](#)
- msgdisc** [34](#), [45](#)
 - description [44](#)
- msgdisc.sql* table creation script [46](#)
- msgdisc.status** [30](#)
 - checking [25](#)
 - KILLED [21](#), [25](#)
 - possible states [22](#)
 - RECEIVED [21](#), [25](#), [34](#)
 - RUNNING [25](#), [34](#)
 - STANDBY [25](#)
- msgftp_TSD/* [38](#)
- msgs.par* [42](#)

N

- no data [32](#)
- no data available [33](#)

O

- ORACLE requirements [14](#)
- outage** [45](#)
 - description [44](#)

P

- ParseData* [6](#), [14](#), [30](#)
 - active process limit [33](#)
 - blocking queue [22](#)
 - error messages [32](#)
 - gradual shutdown [20](#)
 - immediate shutdown [20](#)
 - log files [29](#)
 - reconfiguring [24](#)
 - verify that running [33](#)
- parse error [32](#)
- passwords [25](#)
- performance of Message Subsystem [9](#)
- problem messages, removing [23](#)
- problems
 - reporting [35](#)
- processing, independent [23](#)
- process.par* [41](#)

R

- RECEIVED [25](#)
- request** [45](#)
- RequestFlow* [14](#), [30](#)
- request.state**
 - done-success [30](#)
- run_MessageStore* [39](#)
- RUNNING [25](#)

▼ Index

S

script files location [40](#)
security [25](#)
SEL# database accounts [43](#)
sendmail [6](#), [18](#)
shared.par [34](#), [41](#)
shutdown
 gradual [19](#)
 immediate [20](#)
software
 configuration [23](#)
 environment [14](#)
STANDBY [25](#)

T

technical terms [vi](#)
TSD [vi](#)
TSD FTP [vi](#)
typographical conventions [vi](#)

U

user environment [40](#)

V

verifying active processes [28](#)

W

Workflow [11](#), [14](#)

X

xtag
 description [44](#)